Transport and Chemical Reaction: Characteristics Method and Splitting Methods

Jürgen Geiser and Asgar Jamneshan

November 11, 2008

Abstract

In this paper we present computational schemes for solving transport and chemical reaction processes. The model is based on a convection-diffusion-reaction equation which models our fluid dynamics of the deposition process. For the spatial discretization we combine characteristics methods for the advection equations and use finite volume methods for the diffusion equation. The reaction parts are solved analytically and the solutions are embedded into the spatial discretized advection parts. We present decomposition methods in time to couple the different scales of the multi-component transport equations. The different time-scales are coupled by transfer-operators and we discuss a multi-scale solver. In the numerical experiments we present a microscopic model for partical transport. The accuracy of the methods are discussed.

Keywords: Chemical vapor deposition, multi-scale problem, characteristics method, splitting method.
AMS subject classifications. 35K25, 35K20, 74S10, 70G65.

1 Introduction

We motivate our studies by replacing high-costly real-life simulations by numerical expreminets for low-temperature, low-pressure plasma that can be found in CVD (chemical vapor deposition) processes. Recently, the industrial production of high-temperature films by deposition of low-pressure, low-temperature processes have increased. The interest in standard applications to TiN and TiC are immense. However recently, deposition with new material classes known as MAX-phases become important, too. The MAX-phase are nanolayered terniar metal-carbides or -nitrids, where M is a transition metal, A is an A-group element (e.g. Al, Ga, In, Si, etc.) and X is C (carbon) or N (nitrid). In this paper we discuss a microscopic model that take into account the particle's transport and kinetics. We apply mass-conservation of particles, see [24]. From the mathematical point of view we combine characteristics methods with decomposition methods and resolve each particle with respect to the transport and the reaction scale, see [1], [13].

This paper is outlined as follows. In section 2 we present our mathematical model and a possible reduced model for further approximations. In section 3 we discuss the time and spatial discretization methods. The splitting methods are discussed in section 4; here we also present the coupling ideas for different codes. The numerical experiments are given in Section 5. In Section 6 we summarize our results.

2 Mathematical Model

We will concentrate on conservation of mass and will assume that energy and momentum is conserved, see [10]. Therefore the continuum flow and the kinetics can be described as a convection-diffusion-reaction equation given as:

$$\partial_t \ c_i + \nabla \cdot (\mathbf{v_i} c_i - D_i \nabla c_i) = -\lambda_{ii} c_i + \sum_{k=1, k\neq i}^{m} \lambda_{ik} c_k \ , \tag{1}$$

$$i = 1, \ldots, m \ ,$$

$$c_i(x,t) = c_{i,0}(x), \text{ on } \Omega, \tag{2}$$

$$c_i(x,t) = c_{i,1}(x,t), \text{ on } \partial\Omega \times [0,T], \tag{3}$$

where $c = (c_1, \ldots, c_m)^t$ is the molar concentration of the species and $\mathbf{v_i} = (v_{i1}, \ldots, v_{im})^t$ the flux of the species. $D_i \in \mathbb{R}^m \times \mathbb{R}^m$ is the constant diffusivity matrix. The kinetic term is given by the reaction-rates $\lambda_{11} \ldots, \lambda_{mm}$. The initial value is given as $c_{i,0} = (c_{1,0}, \ldots, c_{m,0})^t$ and we assume a Dirichlet Boundary with a function $c_1(x,t)$ sufficiently smooth.

3 Discretization and Solver methods

In what follows we discuss spatial discretization methods for one particle and for multi-particle equations that are modeled with advection and a system of advection equations respectively.

3.1 Finite volume methods

To solve the remaining advection-reaction part of (26) by finite volume methods, we consider a mesh of nonempty, non-intersecting finite volumes $\Omega_j \subset \Omega$ that cover Ω. We assume that Ω_j, $j = 1, \ldots, N$ are polygonal (consequently, Ω must be polygonal, too).

To simplify the notation, we concentrate on (26) and we write:

$$(\partial_t c_i + \lambda_{ii} c_i) + \nabla \cdot (\boldsymbol{v_i} c_i) = Q \, , \tag{4}$$

where $Q = \sum_{k=1, k\neq i}^{m} \lambda_{ik} c_k$.

Integrating (4) over Ω_j and some time interval (t^n, t^{n+1}), we obtain

$$\int_{\Omega_j} c_i(t^{n+1}) = \int_{\Omega_j} c_i(t^n) - \int_{t^n}^{t^{n+1}} \int_{\partial\Omega_j} \boldsymbol{n}_j \cdot \boldsymbol{v_i}\, c_i + \int_{t^n}^{t^{n+1}} \int_{\Omega_j} (Q - \lambda_{ii} c_i) \ , \qquad (5)$$

where $\boldsymbol{n}_j$ is the unit normal vector with respect to the boundary $\partial\Omega_j$ of Ω_j. For simplicity we skipped the integration variables in (5).

Finally, we denote by c_j^n the average concentration of u at $t = t^n$ in Ω_j, i.e.

$$c_{ij}^n := \frac{1}{|\Omega_j|} \int_{\Omega_j} c_i(t^n, x)\, dx \,, \qquad (6)$$

where $|\Omega_j|$ denotes the volume of Ω_j. We can define the average values c_j^{n+1} in the same manner as we did in (6).

Using the assumptions and notations from above, we can rewrite (5) in a discrete form:

$$|\Omega_j|\, c_{ij}^{n+1} = |\Omega_j|\, c_{ij}^n - \sum_k \int_{t^n}^{t^{n+1}} \int_{\Gamma_{jk}} \boldsymbol{n}_j \cdot \boldsymbol{v_i}\, c_i + \int_{t^n}^{t^{n+1}} \int_{\Omega_j} (Q - \lambda_{ii} c_i) \ , \qquad (7)$$

where the index k is considered only for neighbors Ω_k of Ω_j with common surfaces, i.e. $\Gamma_{jk} := \partial\Omega_j \cap \partial\Omega_k$. Note that here the subscripts j and k are related to the finite volume mesh.

Remark 1 *We have discretized the advaction-reaction equation with respect to the transport model in a conservative form, e.g. to respect the local mass balance. This discretization method can be reformulated in charactersitics methods and we can add the reaction term as a further function, see [18].*

3.2 Numerical solution of the advection equation

If no reactions are considered in (4), the remaining i-th advection equation can be discussed independently and we have the following form:

$$\partial_t c_i + \nabla \cdot (\boldsymbol{v_i} c_i) = 0 \,, \qquad (8)$$

$$i = 1, \ldots, m. \qquad (9)$$

The initial conditions are given by (26), and $u(t, \gamma)$ is explicitly given for $t > 0$ at the inflow boundary $\gamma \in \partial^{in}\Omega$ by (26).

The exact solution of (8) can be directly found using the so-called *forward tracking* form of characteristic curves. If the solution of (8) is known at some time point $t_0 \geq 0$ and some point $y \in \Omega \cup \partial^{in}\Omega$, then u remains constant for

$t \geq t_0$ along the characteristic curve $X = X(t)$, i.e. $u(t, X(t)) = u(t_0, y)$ and

$$X(t) = X(t; t_0, y) = y + \int_{t_0}^{t} \boldsymbol{v_i}(X(s))\, ds\,, \tag{10}$$

$$i = 1, \ldots, m. \tag{11}$$

The characteristic curve $X(t)$ starts at time $t = t_0$ from the point y, i.e. $X(t_0; t_0, y) = y$, and it is forward in time for $t > t_0$. Of course, we may find that $X(t) \notin \Omega$, i.e. the characteristic curve may leave the domain Ω through $\partial^{out}\Omega$.

Consequently, we have $c_i(t, X(t; t_0, y)) = C_i(t_0, y)$, where the function $C_i(0, y)$ is given for $t_0 = 0$ and $y \in \Omega$ by the initial conditions in (26), and for $t_0 > 0$ and $y \in \partial^{in}\Omega$ by the inflow boundary conditions (26).

The solution $c_i(t, x)$ of (8) can also be expressed in a "backward tracking" form which is more suitable for a direct formulation of the discretization schemes. Consequently, for any characteristic curve $X = X(t) = X(t; s, Y)$ that is defined in a forward manner, i.e. $X(s; s, Y) = Y$ and $t \geq s$, we obtain the curve $Y = Y(s) = Y(s; t, x)$, which is defined in a backward manner, i.e. $Y(t; t, X) = X$ and $s \leq t$. If we express Y as function of t_0 for $t_0 \leq t$, we obtain from (10):

$$Y(t_0) = Y(t_0; t, x) = x - \int_{t_0}^{t} \boldsymbol{v_i}(X(s))\, ds\,, \tag{12}$$

and we have $c_i(t, x) = c_i(t_0, Y(t_0))$.

To simplify our treatment of the inflow boundary conditions, we suppose that $U(t, \gamma) = U^{n+1/2} \equiv const$ for $\gamma \in \partial^{in}\Omega$ and $t \in [t^n, t^{n+1})$. Moreover, we formally define for any $\gamma \in \partial^{in}\Omega$ and $t_0 \in [t^n, t^{n+1}]$ $Y(s; t_0, \gamma) \equiv Y(t_0; t_0, \gamma)$ for $t^n \leq s \leq t_0$. Using (7), the standard FVM for differential equation (8), we can reformulate the characteristics method as, see [23]:

$$|\Omega_j|\, c_{ij}^{n+1} \;=\; |\Omega_j|\, c_{ij}^{n} \;-\; \sum_k \int_{t^n}^{t^{n+1}} \int_{\Gamma_{jk}} \boldsymbol{n}_j(\gamma) \cdot \boldsymbol{v_i}(\gamma)\, c_i(t, \gamma)\, d\gamma dt \;\;, \tag{13}$$

$$i = 1, \ldots, m. \tag{14}$$

A flux-based method of characteristics, see [23], applies the substitution $c_i(t, \gamma) = c_i(t^n, Y(t^n; t, \gamma))$ to (13).

For the integration variable $t \in (t^n, t^{n+1})$ and for each point $\gamma \in \partial^{out}\Omega_j$ the characteristic curves $Y(s)$ are tracked backward starting at γ at $s = t$ and ending at $s = t^n$. We must reach a point $Y = Y(t^n)$ such that $Y \in \partial^{in}\Omega$ or $Y \in \Omega$. In the first case $u(t^n, Y)$ is given by the inflow concentration $U(t^n, Y) = U^n$; in the latter it is given by $u(t^n, Y)$.

The integral in the right-hand side of (13) can be solved exactly for the one-dimensional case with general initial and boundary conditions, see [23]. For the

general 2D or 3D case a numerical approximation of $u(t_0, Y(t_0))$ with respect to $Y(t_0)$ shall be used.

3.3 Analytical Solution to a system of ordinay differential equations of first order

In this section we deal with the reaction part of equation 26.

The following ODE system is given:

$$\frac{dc}{dt} = \Phi\ c(t)\ +\ b(t)\ , \tag{15}$$

where $c(t = t_0) = c_0$.

We assume the matrix Φ to be constant and non-singular. Furthermore we assume the matrix to be a M-matrix in order to obtain positive and real eigenvalues which are necessary to obtain real eigenvectors, see [16] and [17].

The matix Φ is given as:

$$\Phi = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1m} \\ \lambda 21 & \lambda_{22} & \dots & \lambda_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1} & \lambda_{m2} & \dots & \lambda_{mm} \end{pmatrix}, \tag{16}$$

Furthermore we assume $b(t)$ to be a smooth function of t.

We solve the equation (15) analytically by using an eigenvalue tranformation of the matrix (16). This method decouples the equations in m independent parts which can be integrated separately.

We have

$$c(t) = c_{homo}(t) + c_p(t)\ , \tag{17}$$

where $c(t = t_0) = c_0$.

For the homogeneous part we have:

$$c_{homo}(t) = W_c \exp(\Lambda(t - t_0))\tilde{c}\ , \tag{18}$$

where $\tilde{c} = W_c^{-1} c_0$, Λ is the eigenvalue matrix and

$$\exp(\Lambda(t - t_0)) = \begin{pmatrix} \exp(\lambda_1(t - t_0)) & 0 & \dots & 0 \\ 0 & \exp(\lambda_2(t - t_0)) & \dots & 0 \\ & & \ddots & \\ 0 & 0 & & 0 \\ 0 & 0 & \dots & \exp(\lambda_n(t - t_0)) \end{pmatrix}. \tag{19}$$

For the inhomogeneous part we have:

$$c_p(t) = W_c \exp(\Lambda(t))u(t)\ , \tag{20}$$

where $u(t) = \int_{t_0}^{t} (\exp(\Lambda(t)))^{-1} W_c^{-1} b(t) dt$ and the integration can be approximated with a numerical integration method, e.g. [25].

We obtain the following solution:

$$\begin{aligned} c(t) &= W_c \exp(\Lambda(t - t_0)) W_c^{-1} c_0 \\ &+ W_c \exp(\Lambda(t)) u(t) \ , \end{aligned} \tag{21}$$

where c_0 is the initial condition.

Remark 2 *The solution is useful if we have non-singular matrices or if the reactants have a successor. Otherwise we obtain fast numerical solvers.*

3.4 Numerical Methods for ODEs

Here we introduce numerical methods which we apply to solve the singular reaction matrix of the underlying ODEs with $det(\Phi) = 0$.

We use the following methods:

We use the implicit trapezoidal rule

$$\begin{array}{c|cc} 0 & & \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{22}$$

Furthermore, we use the following implicit Runge-Kutta methods:

Lobatto IIIA

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & -\frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & -\frac{1}{6} \end{array} \tag{23}$$

Remark 3 *We can also apply integration methods for the right hand side.*

3.5 Computation of mass conservation

It is important that the numerical methods do not violate physical laws, see [19].

In our special model this signifies that the mass of the gaseous concentration in the apparatus has to be conserved during the transport, see [18]. To control this behaviour, we have to calculate the following equation:

$$\sum_{i=1}^{n} \int_{\Omega} c_i(s,t) \ , \tag{24}$$

where n is the number of species, Ω is the spatial-domain and $t \in [0, T]$ is a predefined time-point.

To obtain the total mass through all time-intervalls, we have to calculate:

$$\sum_{i=1}^{n} \int_0^T \int_\Omega c_i(s,t)\, ds\, dt, \tag{25}$$

where n is the number of species, Ω is the spatial-domain and $[0,T]$ is the time domain.

Often the physical behaviour of higher moments with respect to the mass is important for the conservation and we have to compute:

$$\sum_{i=1}^{n} \int_\Omega s\, c_i(s,t)\ , \tag{26}$$

where n is the number of species, Ω is the spatial-domain and $t \in [0,T]$ is a predefined time-point.

Remark 4 *The finite volume and finite difference methods conserve mass, see [18]. The analytical computation and the operator splitting methods, too.*

4 Splitting Methods

4.1 Splitting methods of first order for linear equations (A-B-splitting)

First we describe the simplest operator-splitting, the so-called *sequential splitting*, for the following system of ordinary linear differential equations:

$$\partial_t c(t) = A\, c(t) \ +\ B\, c(t)\ ,\quad t \in (0,T),\quad c(0) = c_0, \tag{27}$$

where the initial value c_0 is given and A and B are assumed to be bounded operators in a Banach-space. For example, we could also discretize the spatial variable and get operators (matrices) in an ODE context. Hence, they can be considered as bounded operators.

The sequential operator-splitting method is introduced as a method for solving two sub-problems sequentially on sub-intervals $[t^n, t^{n+1}]$, where $n = 0, 1, \ldots, N-1$ and $t^N = T$. The different sub-problems are connected via the initial conditions. This means that we replace the original problem (27) with the sub-problems

$$\begin{aligned} \frac{\partial c^*(t)}{\partial t} &= Ac^*(t)\ , \quad \text{with } c^*(t^n) = c^n\ , \\ \frac{\partial c^{**}(t)}{\partial t} &= Bc^{**}(t)\ , \quad \text{with } c^{**}(t^n) = c^*(t^{n+1})\ , \end{aligned} \tag{28}$$

for $n = 0, \ldots, N-1$, where $c^0 = c_0$ is given from (27). The approximated split solution at the point $t = t^{n+1}$ is defined as $c^{n+1} = c^{**}(t^{n+1})$.

Clearly, the change of the original problem with the sub-problems usually results in some error, called *splitting error*. The splitting error of the sequential splitting method can be derived as follows (cf. e.g.[8]):

$$\begin{aligned}\rho_n &= \frac{1}{\tau}(\exp(\tau_n(A+B)) - \exp(\tau_n B)\exp(\tau_n A))\ c(t^n) \\ &= \frac{1}{2}\tau_n[A,B]\ c(t^n) + O(\tau_n^2)\ . \end{aligned} \tag{29}$$

where the splitting time-step is defined as $\tau_n = t^{n+1} - t^n$ and $[A,B] := AB - BA$ is the commutator of A and B. Consequently, the splitting error is $O(\tau_n)$ when the operators A and B do not commute, otherwise the method is exact. Hence, by definition, the sequential splitting is called *first order splitting method*.

In the next subsection we present the iterative-splitting method.

4.2 Iterative splitting method

The following algorithm is based on the iteration with fixed splitting discretization step-size τ. On the time interval $[t^n, t^{n+1}]$ we solve the following sub-problems consecutively for $i = 0, 2, \ldots 2m$. (cf. [14] and [6].):

$$\frac{\partial c_i(t)}{\partial t} = Ac_i(t) \ + \ Bc_{i-1}(t), \text{ with } \ c_i(t^n) = c^n \tag{30}$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = Ac_i(t) \ + \ Bc_{i+1}(t), \text{ with } \ c_{i+1}(t^n) = c^n\ , \tag{31}$$

where $c_0(t^n) = c^0$, $c_{-1} = 0$ and c^n is the known split approximation at time-level $t = t^n$. The split approximation at time-level $t = t^{n+1}$ is defined as $c^{n+1} = c_{2m+1}(t^{n+1})$. (Clearly, the function $c_{i+1}(t)$ depends on the interval $[t^n, t^{n+1}]$ too, but, for the sake of simplicity, we omit the dependence on n in our notation.)

In the following we will analyze the convergence and the rate of convergence of the method (30)–(31) for m tends to infinity for the linear operators $A, B : \mathbf{X} \to \mathbf{X}$, where we assume that these operators and their sum are generators of the C_0 semi-groups. We emphasize that these operators aren't necessarily bounded so the convergence is examined in a general Banach space setting.

4.3 Iterative splitting method with embedded Multigrid methods

The following algorithm is based on embedding the multigrid method in the operator splitting method. The iteration is done with fixed splitting discretization step-size τ. On the time interval $[t^n, t^{n+1}]$ we solve the following sub-problems consecutively for $i = 0, 2, \ldots 2m$. (cf. [14] and [6].):

$$\frac{\partial c_i(t)}{\partial t} = Ac_i(t) \; + \; P^{l_A - l_B} B c_{i-1}(t), \text{ with } \; c_i(t^n) = c^n \tag{32}$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = R^{l_A - l_B} A c_i(t) \; + \; B c_{i+1}(t), \text{ with } \; c_{i+1}(t^n) = c^n \; , \tag{33}$$

where $c_0(t^n) = c^n \; , \; c_{-1} = 0$ and c^n is the known split approximation at time-level $t = t^n$. We assume A to be the fine spatial discretized operator on level l_A and B to be the coarse discretized operator on level l_B.

The operators are coupled by the restriction and prolongation operators:

$$A_{coarse} = R^{l_A - l_B} A, \tag{34}$$

$$B_{fine} = P^{l_A - l_B} B, \tag{35}$$

where R is the restriction and P the prolongation operator.

Theorem 1 *Let us consider the abstract Cauchy problem in a Banach space* $\mathbf{X}$

$$\begin{gathered} \partial_t c(t) = Ac(t) + P^{l_A - l_B} B c(t), \quad 0 < t \leq T \\ c(0) = c_0 \end{gathered} \tag{36}$$

where $A, P^{l_A - l_B} B, A + P^{l_A - l_B} B \; : \mathbf{X} \; \rightarrow \mathbf{X}$ *are given linear operators being generators of the* C_0*-semi-group and* $c_0 \in \mathbf{X}$ *is a given element. Then the iteration process (32)–(33) is convergent and the rate of convergence is of higher order.*

Proof 1 *We assume* $A + P^{l_A - l_B} B \in \mathcal{L}(X)$ *and assume a generator of a uniformly continuous semi-group, hence the problem (36) has a unique solution* $c(t) = \exp((A + P^{l_A - l_B} B)t)c_0$.

Let us consider the iteration (30)–(31) on the sub-interval $[t^n, t^{n+1}]$. *For the local error function* $e_i(t) = c(t) - c_i(t)$ *we have the relations*

$$\begin{gathered} \partial_t e_i(t) = Ae_i(t) + P^{l_A - l_B} B e_{i-1}(t), \quad t \in (t^n, t^{n+1}], \\ e_i(t^n) = 0, \end{gathered} \tag{37}$$

and

$$\begin{gathered} \partial_t e_{i+1}(t) = R^{l_A - l_B} A e_i(t) + B e_{i+1}(t), \quad t \in (t^n, t^{n+1}], \\ e_{i+1}(t^n) = 0, \end{gathered} \tag{38}$$

for $m = 0, 2, 4, \ldots$, *with* $e_0(0) = 0$ *and* $e_{-1}(t) = c(t)$. *In the following we use the notations* $\mathbf{X}^2$ *for the product space* $\mathbf{X} \times \mathbf{X}$ *endowed with the norm* $\|(u, v)\| = \max\{\|u\|, \|v\|\}$ $(u, v \in \mathbf{X})$. *The elements* $\mathcal{E}_i(t), \; \mathcal{F}_i(t) \in \mathbf{X}^2$ *and the linear operator* $\mathcal{A} : \mathbf{X}^2 \rightarrow \mathbf{X}^2$ *are defined as follows*

$$\mathcal{E}_i(t) = \begin{bmatrix} e_i(t) \\ e_{i+1}(t) \end{bmatrix}, \quad \mathcal{F}_i(t) = \begin{bmatrix} P^{l_A - l_B} B e_{i-1}(t) \\ 0 \end{bmatrix}, \quad \mathcal{A} = \begin{bmatrix} A & 0 \\ R^{l_A - l_B} A & B \end{bmatrix}. \tag{39}$$

Then using the notations (39), the relations (37) and (38) can be written in the form

$$\begin{aligned} \partial_t \mathcal{E}_i(t) = \mathcal{A}\mathcal{E}_i(t) + \mathcal{F}_i(t), \quad t \in (t^n, t^{n+1}], \\ \mathcal{E}_i(t^n) = 0. \end{aligned} \tag{40}$$

Due to our assumptions, $\mathcal{A}$ is a generator of the one-parameter C_0-semi-group $(\exp \mathcal{A}t)_{t\geq 0}$, hence using the variations of constants formula, the solution to the abstract Cauchy problem (40) with homogeneous initial condition can be written as

$$\mathcal{E}_i(t) = \int_{t^n}^{t} \exp(\mathcal{A}(t-s))\mathcal{F}_i(s)ds, \quad t \in [t^n, t^{n+1}]. \tag{41}$$

(See, e.g. [4].) Hence, using the denotation

$$\|\mathcal{E}_i\|_\infty = \sup_{t\in[t^n,t^{n+1}]} \|\mathcal{E}_i(t)\| \ , \tag{42}$$

we have

$$\begin{aligned} \|\mathcal{E}_i\|(t) \leq \|\mathcal{F}_i\|_\infty \int_{t^n}^{t} \|\exp(\mathcal{A}(t-s))\|ds \\ = \|B\|\|e_{i-1}\| \int_{t^n}^{t} \|\exp(\mathcal{A}(t-s))\|ds, \quad t \in [t^n, t^{n+1}]. \end{aligned} \tag{43}$$

Since $(\mathcal{A}(t))_{t\geq 0}$ is a semi-group, the so called growth estimation

$$\|\exp(\mathcal{A}t)\| \leq K \exp(\omega t), \quad t \geq 0, \tag{44}$$

holds with some numbers $K \geq 0$ and $\omega \in \mathbb{R}$, cf. [4].

- *Assume that $(\mathcal{A}(t))_{t\geq 0}$ is a bounded or a exponentially stable semi-group, i.e. (44), holds with some $\omega \leq 0$. Then obviously the estimate*

  $$\|\exp(\mathcal{A}t)\| \leq K, \quad t \geq 0, \tag{45}$$

  holds, and hence, on base of (43) we have the relation

  $$\|\mathcal{E}_i\|(t) \leq K\|P^{l_A - l_B} B\|\tau_n\|e_{i-1}\|, \quad t \in [t^n, t^{n+1}]. \tag{46}$$

- *Assume that $(\exp \mathcal{A}t)_{t\geq 0}$ has an exponential growth with some $\omega > 0$. Using (44), we have*

  $$\int_{t^n}^{t} \|\exp(\mathcal{A}(t-s))\|ds \leq K_\omega(t), \quad t \in [t^n, t^{n+1}], \tag{47}$$

  where

  $$K_\omega(t) = \frac{K}{\omega} \left(\exp(\omega(t-t^n)) - 1\right), \quad t \in [t^n, t^{n+1}]. \tag{48}$$

  Hence

  $$K_\omega(t) \leq \frac{K}{\omega} \left(\exp(\omega\tau_n) - 1\right) = K\tau_n + \mathcal{O}(\tau_n^2). \tag{49}$$

The estimations (46) and (49) result in

$$\|\mathcal{E}_i\|_\infty = K\|P^{l_A - l_B}\|\|B\|\tau_n\|e_{i-1}\| + \mathcal{O}(\tau_n^2). \tag{50}$$

Taking into account the definition of $\mathcal{E}_i$ *and the norm* $\|\cdot\|_\infty$*, we obtain*

$$\|e_i\| = K\|P^{l_A - l_B}\|\|B\|\tau_n\|e_{i-1}\| + \mathcal{O}(\tau_n^2), \tag{51}$$

and hence

$$\|e_{i+1}\| = K_1\tau_n^2\|e_{i-1}\| + \mathcal{O}(\tau_n^3), \tag{52}$$

which proves our statement.

In the next subsection we present the used time-discretization methods.

4.4 Prolongation and Restriction Operators

Transfer operators are used in multigrid methods to interpolate the solutions between each grid. They can be generated on the basis of usual central formulas of weighting and linear interpolation to restriction and prolongation. In the quite natural elliptic problems with smooth coefficients it can be done by [12]. It is not so evident in our case with hyprerbolic equations where discontinuous solutions with shocks arise often, see [21].

For such problems we apply the linear interpolation operators:

$$\begin{aligned} P &= 1/2[121]^t, \\ R &= 1/4[121], \end{aligned}$$

A more stable process can be applied by the mixed contral upwind-transfer where we take into account the spectrum of the matrices:

$$\begin{aligned} P : f_i &= F_{i/2}, \\ f_{i+1} &= \beta_i F_{i/2} + (1-\beta_i)F_{i/2+1}, \\ \beta_i &= (q_{i+1}(q_{i+2} - q_i) + 1)/2, \\ R : F_{i/2} &:= \alpha_i f_{i-1} + (1-\alpha_i)f_i + \alpha_i f_{i+1}, \\ \alpha_i &= (1-q_i)/4, \end{aligned}$$

where

$$\begin{aligned} P : f_i &= F_{i/2}, \\ f_{i+1} &= \beta_i F_{i/2} + (1-\beta_i)F_{i/2+1}, \\ \beta_i &= (q_{i+1}(q_{i+2} - q_i) + 1)/2, \\ R : F_{i/2} &:= \alpha_i f_{i-1} + (1-\alpha_i)f_i + \alpha_i f_{i+1}, \\ \alpha_i &= (1-q_i)/4, \end{aligned}$$

$$
\begin{aligned}
q_{i-1} = q_i = 1, \text{ if } CFL_i, CFL_{i-1} \leq 1, \text{and } c_{i-1} > c_i + \epsilon \\
q_{i-1} = q_i = 1, \text{ if } CFL_i, CFL_{i-1} \geq 1, \text{and } c_i > c_{i-1} + \epsilon \\
q_i = 0, \text{ otherwise} \\
q_0 = q_I = 1,
\end{aligned}
$$

where q_i is our shock indicator function and is based on the CFL condition. If we are on a shock, we apply 1, otherwise we have 0.

4.5 Iterative Solvers: Waveform Relaxation

A method to solve large coupled differential equations is the waveform relaxation scheme.

The iterative method was discussed in [26], [15] and [11]. There exist Gauss- or Jacobian schemes for the method which help to decouple the schemes more or less effective.

We deal with the following ordinary differential equation or assume a semi-discretized partial differential equation:

$$
\begin{aligned}
u_t &= f(u,t), \text{ in } (0,T)\ , \\
u(0) &= v_0
\end{aligned}
$$

where $u = (u_1, \ldots, u_m)^t$ and $f(u,t) = (f_1(u,t), \ldots, f_m(u,t))^t$.

We apply the Waveform-Relaxation method for $i = 0, 1, \ldots m$ and obtain:

$$
\begin{aligned}
&\frac{\partial u_{1,i}(x,t)}{\partial t} = f_1(u_{1,i}, u_{2,i-1}, \ldots, u_{m,i-1}) \text{ with } u_{1,i}(t^n) = u_1(t^n) && (53)\\
&\frac{\partial u_{2,i}(x,t)}{\partial t} = f_2(u_{1,i-1}, u_{2,i}, u_{3,i-1} \ldots, u_{m,i-1}) \text{ with } u_{2,i}(t^n) = u_2(t^n) && (54)\\
&\vdots \\
&\frac{\partial u_{m,i}(x,t)}{\partial t} = f_2(u_{1,i-1}, \ldots, u_{m-1,i-1}, u_{m,i}) \text{ with } u_{m,i}(t^n) = u_m(t^n) && (55)
\end{aligned}
$$

where for the initialization of the first step we have $u_{1,-1}(t) = u_1(t^n), \ldots, u_{m,-1}(t) = u_m(t^n)$.

We reduce the system of equations to a system of two equations and the method is reformulated in our iterative splitting methods.

So we deal with:

$$
\begin{aligned}
\frac{\partial u_1}{\partial t} &= f_{11}(u_1,t) + f_{12}(u_2,t), \text{ in } (0,T)\ , && (56)\\
\frac{\partial u_2}{\partial t} &= f_{21}(u_1,t) + f_{22}(u_2,t), \text{ in } (0,T)\ , && (57)\\
u(0) &= v_0 && (58)
\end{aligned}
$$

where $u = (u_1, u_2)^t$.

We have the following operator equation:

$$\frac{\partial u}{\partial t} = A(u) + B(u), \text{ in } (0,T)\,, \tag{59}$$

$$u(0) = v_0 \tag{60}$$

where the operators are given as

$$A(u) = \begin{pmatrix} f_{11}(u_1) \\ f_{21}(u_1) \end{pmatrix} \tag{61}$$

$$B(u) = \begin{pmatrix} f_{12}(u_2) \\ f_{22}(u_2) \end{pmatrix} \tag{62}$$

The iterative splitting method as Waveform-Relaxation method is written for $i = 0, 1, \ldots m$ as:

$$\frac{\partial u_i}{\partial t} = A(u_{1,i}, u_{2,i-1}) + B(u_{1,i-1}, u_{2,i}) \tag{63}$$

$$\text{with } \; u_{1,i}(t^n) = u_1(t^n) \text{ and } u_{2,i}(t^n) = u_2(t^n)$$

where for the initialization of the first step we have $u_{1,-1}(t) = u_1(t^n), u_{2,-1}(t) = u_2(t^n)$.

5 Numerical Experiments: Microscopic models

In this section we deal with the microscopic models. We deal in a first model with the pure reaction case of the heavy particles and in a next model with the transport and reaction of such particles.

5.1 One dimensional particle model: Reaction model

The reaction of the heavy particles are given in the following equations:

$$\partial_t u_1 = -\lambda_1\, u_1 + \lambda_2\, u_2 + \ldots + \lambda_m\, u_m \;\text{ in } \Omega \times (0,T)\,, \tag{64}$$

$$\vdots \tag{65}$$

$$\partial_t u_m = \lambda_1\, u_1 + \lambda_2\, u_2 + \ldots - \lambda_m\, u_m \;\text{ in } \Omega \times (0,T)\,, \tag{66}$$

$$u(0) = (c_1, \ldots, c_m) \;\in \Omega\,, \tag{67}$$

where the initial conditions are given with $u(0)$ and the reversible and irreversible reactions are given with the matrix:

$$\Lambda = \begin{pmatrix} \lambda_{1,1} & \ldots & \lambda_{n,1} \\ \lambda_{1,2} & \ddots & \lambda_{n,2} \\ \vdots & \ddots & \vdots \\ \lambda_{1,n} & \ldots & \lambda_{n,n} \end{pmatrix} \tag{68}$$

$$u(0) = (c_1, \ldots, c_5) \;\in \Omega\,, \tag{69}$$

Reactions with 5 particles

In the next experiment we deal with 5 particles. The equation is given as:

$$\partial_t u = \Lambda u \text{ , in } \times (0,T) \text{ ,} \tag{70}$$
$$u(0) = (c_1, \ldots, c_5) \text{ ,} \tag{71}$$

where we have Λ as reaction matrix of the species.

The parameters are given as:

$$\Lambda = \begin{pmatrix} -(\lambda_{12}+\lambda_{13}+\lambda_{14}+\lambda_{15}) & \ldots & \lambda_{51} \\ \lambda_{12} & \ddots & \lambda_{52} \\ \vdots & \ddots & \vdots \\ \lambda_{15} & \ldots & -(\lambda_{51}+\lambda_{52}+\lambda_{53}+\lambda_{54}) \end{pmatrix} \tag{72}$$
$$u(0) = (c_1, \ldots, c_5) \ \in \Omega \text{ ,} \tag{73}$$

For example:

$$\Lambda = \begin{pmatrix} -1.4 & 0.1 & 0.1 & 0.8 & 0.1 \\ 0.2 & -2.1 & 0.7 & 0.9 & 0.2 \\ 0.3 & 0.8 & -2.6 & 0.1 & 0.3 \\ 0.2 & 0.9 & 0.9 & -2.0 & 0.4 \\ 0.7 & 0.3 & 0.9 & 0.2 & -1.0 \end{pmatrix} \tag{74}$$
$$u(0) = (0.2, 0.2, 0.2, 0.2, 0.2) \tag{75}$$

We have the following results, see Figures 1 and 2:

Reactions with 10 particles

To deal with more realistic reaction schemes we present here the next experiment with 10 particles. The equation is given as:

$$\partial_t u = \Lambda u \text{ , in } \times (0,T) \text{ ,} \tag{76}$$
$$u(0) = (c_1, \ldots, c_{10}) \text{ ,} \tag{77}$$

where we have Λ as reaction matrix of the species.

The parameters are given as:

$$\Lambda = \begin{pmatrix} \lambda_{1,1} & \ldots & \lambda_{10,1} \\ \lambda_{1,2} & \ddots & \lambda_{10,2} \\ \vdots & \ddots & \vdots \\ \lambda_{1,10} & \ldots & \lambda_{10,10} \end{pmatrix} \tag{78}$$
$$u(0) = (c_1, \ldots, c_1 0) \ \in \Omega \text{ ,} \tag{79}$$

For example:

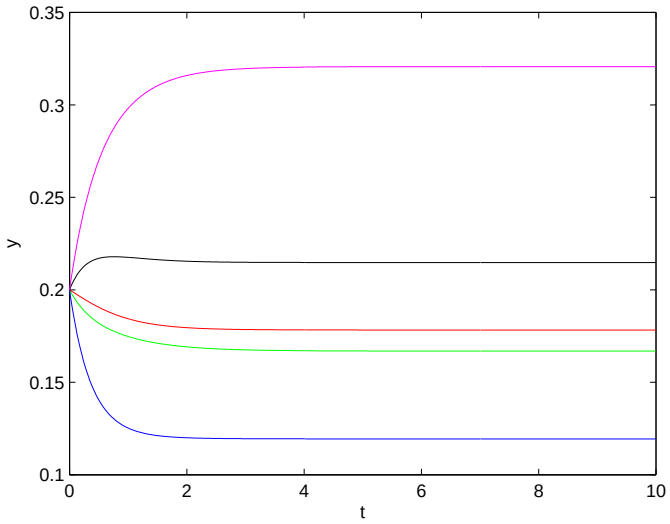

Figure 1: Results of the reaction of 5 species.

$$\Lambda = \begin{pmatrix} -1.7 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & -1.7 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & -1.7 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & -1.7 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & -1.7 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & -1.7 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & -1.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & -1.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & -1.7 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & -0.9 \end{pmatrix} \tag{80}$$

$$u(0) = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1) \tag{81}$$

We have the following results, see Figures 3 and 4:

5.2 Heavy particle Model: Tranport Model

In this model we deal with a transport model of particles. We assume that they are uncoupled and assume different velocities:

$$\partial_t u_1 = -v_1 \, \partial_x u_1 \ \text{ in } \Omega \times (0, T) \, , \tag{82}$$

$$\vdots \tag{83}$$

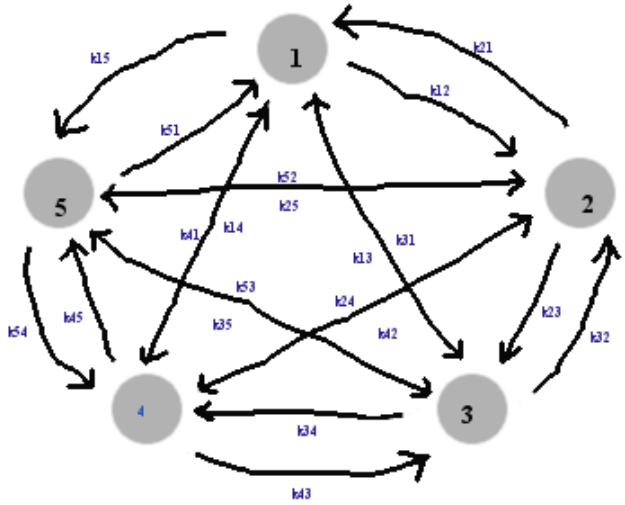

Figure 2: The reaction scheme of the 5 species.

$$\partial_t u_n = -v_n \, \partial_x u_n \quad \text{in } \Omega \times (0,T) \,, \tag{84}$$
$$u(0) = (c_1, \ldots, c_m) \; \in \Omega \,, \tag{85}$$

To solve the uncoupled problem we can apply for each equation a characteristics method. For the separate grids we obtain the best results for the individual CFL conditions:

$CFL = \frac{v_i \Delta t}{\Delta x_i}$

where we assume a fix time-step Δt and obtain the individual mesh-sizes:

$\Delta x_i = v_i \Delta t$

Here we present an example of transport of five particles with the following velocities:

$v_1 = 1, \quad v_2 = \frac{1}{2}, \quad v_3 = \frac{1}{4}, \quad v_4 = \frac{1}{8}, \quad v_5 = \frac{1}{16}$

We obtain the following results, see Figures 5, 6, 7, 8 and 9:

5.3 Heavy-Particle Model (One-dimensional velocity)

For the heavy particle model we have a drift velocity for all the particles in the system and an additionally particle velocity for each particle in the system. We assume an average velocity for the particles in the system.

We have the following particle system:

$$\partial_t u + v \; u = \Lambda u \text{ in } \Omega \times (0,T) \,, \tag{86}$$
$$u_0(x) = u(x,0) \;\; \text{on } \Omega \,, \tag{87}$$
$$u(x,t) = 0 \;\; \text{on } \partial\Omega \times (0,T) \,, \tag{88}$$

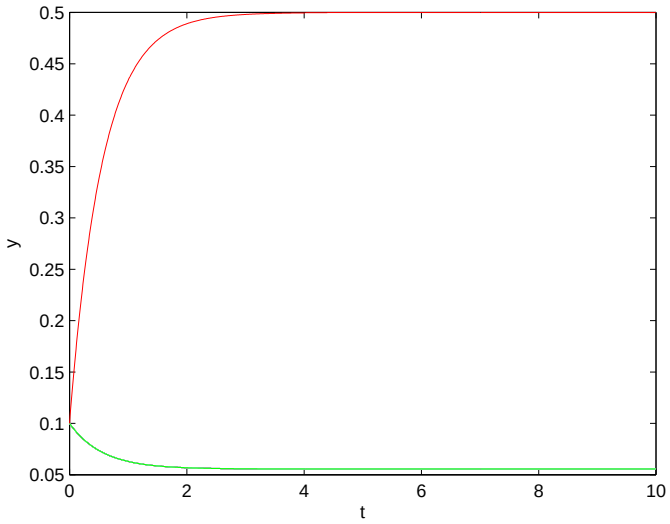

Figure 3: Results of the reaction of 10 species. Example 1.

where $c = c_1, \ldots, m$ is the solution of the transport-reaction equation. The velocity vector is given as $v = (v_1, \ldots, v_m)^t$ and the reaction matrix is given as:

$$\Lambda = \begin{pmatrix} \lambda_{11} & \ldots & \lambda_{m1} \\ \lambda_{12} & \ddots & \lambda_{m2} \\ \vdots & \ddots & \vdots \\ \lambda_{1m} & \ldots & \lambda_{mm} \end{pmatrix}, \tag{89}$$

where the constant reaction rates are $\lambda_{11}, \ldots, \lambda_{mm}$. The analytical solution for the reaction is given in the appendix.

5.3.1 Heavy-Particle Model: Two Particles (One-dimensional velocity)

First we simulate a two component flow:

$v_1 = 1.0, v_2 = 0.5$

The reaction matrix is given as in the appendix. We apply the following splitting method:

$$\frac{\partial c_1(t)}{\partial t} = Ac, \text{ with } c_1(t^n) = c^n \tag{90}$$

$$\frac{\partial c_2(t)}{\partial t} = Bc_2(t), \text{ with } c_2(t^n) = c_1^{n+1}, \tag{91}$$

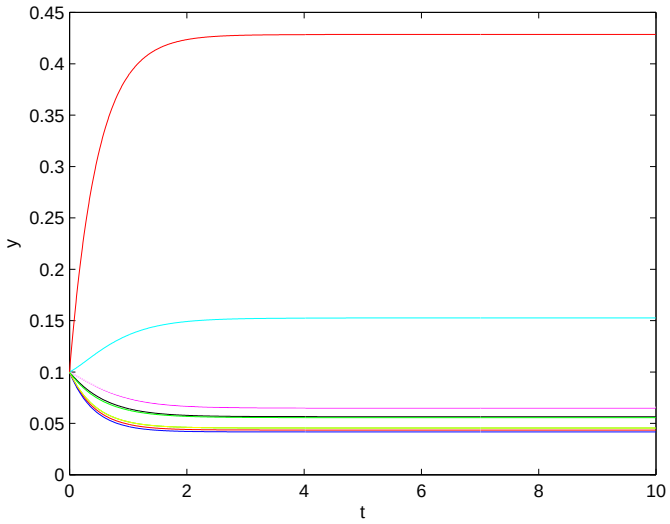

Figure 4: Results of the reaction of 10 species. Example 2

where the matrix A is the spatial discretized matrix and matrix B the reaction matrix. All the spatial discretized matrices are prolongated to the finest scale. We apply the linear prolongation and restriction operators, see 4.

We have the following results, see Figures 10 and 11:

5.3.2 Heavy-Particle Model: Three Particles (One-dimensional velocity)

We simulate a three component flow:

$v_1 = 1.0, v_2 = 0.5, v_3 = 0.25$

The reaction matrix is given as in the appendix. We apply the following splitting method:

$$\frac{\partial c_1(t)}{\partial t} = Ac, \text{ with } c_1(t^n) = c^n \tag{92}$$

$$\frac{\partial c_2(t)}{\partial t} = Bc_2(t), \text{ with } c_2(t^n) = c_1^{n+1}\,, \tag{93}$$

where the matrix A is the spatial discretized matrix and matrix B the reaction matrix. All the spatial discretized matrices are prolongated to the finest scale. We apply the linear prolongation and restriction operators, see 4.

We have the following results; see Figures 12, 13 and 14:

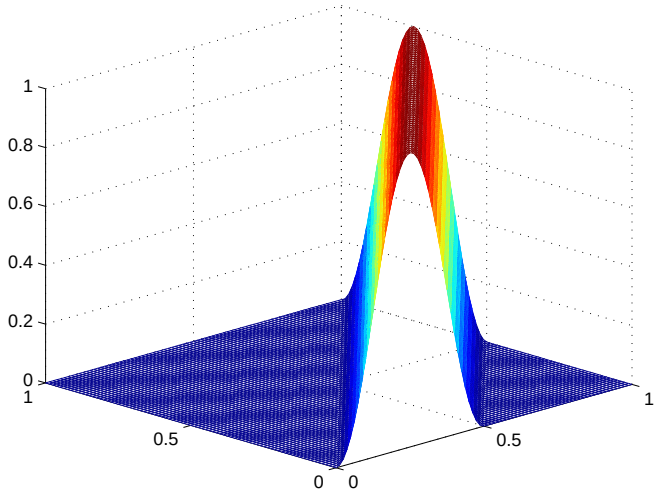

Figure 5: Graph of first particle

6 Conclusions and Discussions

We presented a continuous or kinetic model relative to the fare field or near field effect. Based on the different scale models we could predict the flow of the reacting chemicals on the scale of the chemical reactor. For the mesoscopic scale model we discussed the discretization and solver methods. Numercial examples were presented to discuss the influence of a near-continuum regime at the thin film. The modelling of various inflow sources could describe the growth of the thin-film at the wafer. In future we will analyze the validity of the models with physical experiments.

References

[1] G. Adomian. *Solving Frontier Problems of Physics: The Decomposition Method.* Kluwer Academic Publishers, Dordrecht, 1994.

[2] I. Farago, and Agnes Havasi. *On the convergence and local splitting error of different splitting schemes.* Eötvös Lorand University, Budapest, 2004.

[3] P. Csomós, I. Faragó and A. Havasi. *Weighted sequential splittings and their analysis.* Comput. Math. Appl., (to appear)

[4] K.-J. Engel, R. Nagel, *One-Parameter Semigroups for Linear Evolution Equations.* Springer, New York, 2000.

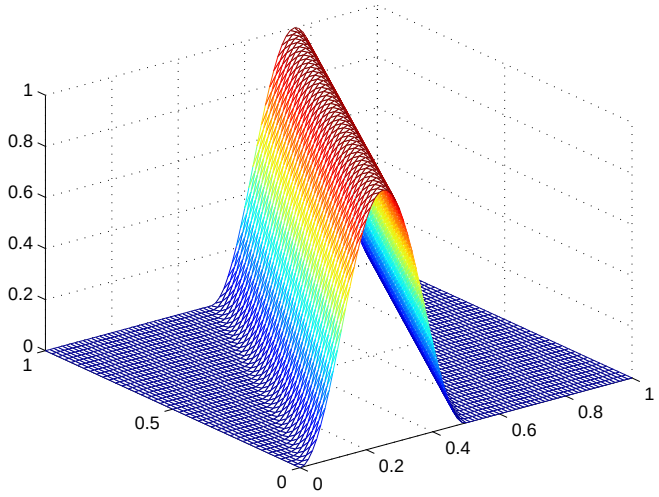

Figure 6: Graph of second particle

[5] I. Farago. *Splitting methods for abstract Cauchy problems.* Lect. Notes Comp.Sci. 3401, Springer Verlag, Berlin, 2005, pp. 35-45

[6] I. Farago, J. Geiser. *Iterative Operator-Splitting methods for Linear Problems.* Preprint No. 1043 of the Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany, June 2005.

[7] J. Geiser. *Numerical Simulation of a Model for Transport and Reaction of Radionuclides.* Proceedings of the Large Scale Scientific Computations of Engineering and Environmental Problems, Sozopol, Bulgaria, 2001.

[8] J. Geiser. *Gekoppelte Diskretisierungsverfahren für Systeme von Konvektions-Dispersions-Diffusions-Reaktionsgleichungen.* Doktor-Arbeit, Universität Heidelberg, 2003.

[9] J. Geiser. *Discretization methods with analytical solutions for convection-diffusion-dispersion-reaction-equations and applications.* Journal of Engineering Mathematics, published online, Oktober 2006.

[10] M.K. Gobbert and C.A. Ringhofer. *An asymptotic analysis for a model of chemical vapor deposition on a microstructured surface.* SIAM Journal on Applied Mathematics, 58, 737–752, 1998.

[11] J. Janssen. *Acceleration of waveform relaxation methods for linear ordinary and partial differential equations.* PhD thesis, K.U.Leuven, Belgium, 1997.

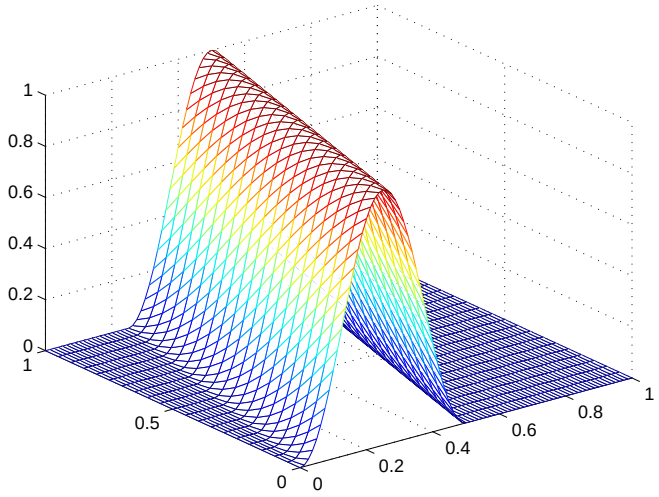

Figure 7: Graph of third particle

[12] W. Hackbusch. *Multi-Gird Methods and Applications.* Springer-Verlag, Berlin, Heidelberg, 1985.

[13] W. Hundsdorfer and J.G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, SpringerVerlag, Berlin, New York, Heidelberg, 2003.

[14] J. Kanney, C. Miller, and C.T. Kelley. *Convergence of iterative split-operator approaches for approximating nonlinear reactive transport problems.* Advances in Water Resources, 26:247–261, 2003.

[15] J. Kaye and A. Sangiovanni-Vincentelli. *Solution of piecewise linear ordinary differential equations using waveform relaxation and Laplace transforms.* Proc. Int. Conf. on Circ. and Comp., New York, 1982.

[16] M. Kamat and D. Keffer. *An analytical theory for diffusion of fluids in crystalline nanoporous materials.* Mol. Phys., 101(10), 1399-1412, 2003.

[17] D. Keffer. *An analytical method for solving systems of linear n-th order ODEs.* self-contained lecture, University of Tennessee, Knoxville, June 1999.

[18] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics , Cambridge University Press, 2002.

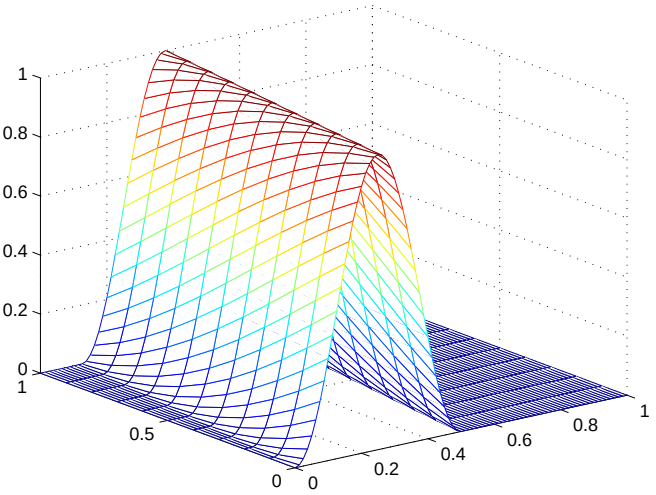

Figure 8: Graph of fourth particle

[19] M.A. Lieberman and A.J. Lichtenberg. *Principle of Plasma Discharges and Materials Processing.* Wiley-Interscience, AA John Wiley & Sons, Inc Publication, Second edition, 2005.

[20] Chr. Lubich. *A variational splitting integrator for quantum molecular dynamics.* Report, 2003.

[21] R.H. Ni. *A Multiple grid scheme for solving the Euler equations.* Proc. AIAA 5th Computational Fluid Dynamics Conf., Palo Alto, Paper No. 81-0132, 257–264, 1981.

[22] M. Ohring. *Materials Science of Thin Films.* Academic Press, San Diego, New York, Boston, London, Second edition, 2002.

[23] P.J. Roache. *A flux-based modified method of characteristics.* Int. J. Numer. Methods Fluids, 12:12591275, 1992.

[24] T.K. Senega and R.P. Brinkmann. *A multi-component transport model for non-equilibrium low-temperature low-pressure plasmas.* J. Phys. D: Appl.Phys., 39, 1606–1618, 2006.

[25] J. Stoer and R. Burlisch. *Introduction to numerical analysis.* Springer verlag, New York, 1993.

[26] S. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems.* Teubner Skripten zur Numerik, B.G. Teubner Stuttgart, 1993.

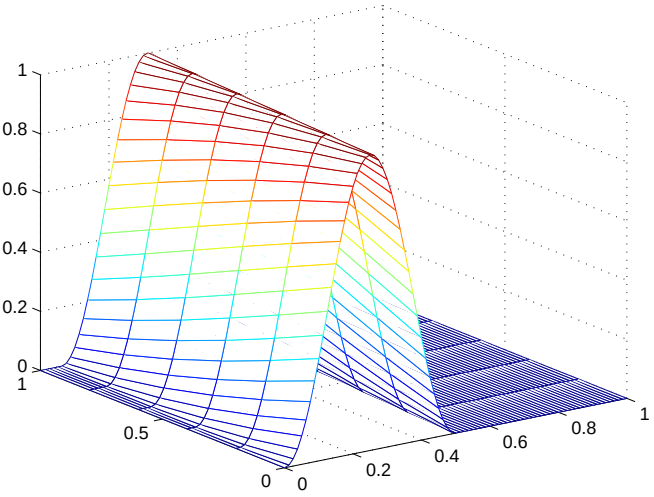

Figure 9: Graph of fifth particle

7 Appendix

7.1 Matlab Commands

In this section we introduce some basic Matlab commands that are useful for solving numerically differential equations that we are dealing with.

Vectors and Matrices

The basic data type in Matlab is an n-dimensional array of double precision numbers. The following commands show how to enter numbers, vectors and matrices, and assign them to variables.

```
>> a = 2
a =

     2
>> x = [1;2;3]
x =

     1
     2
     3
>> A = [1 2 3; 4 5 6; 7 8 9]
```

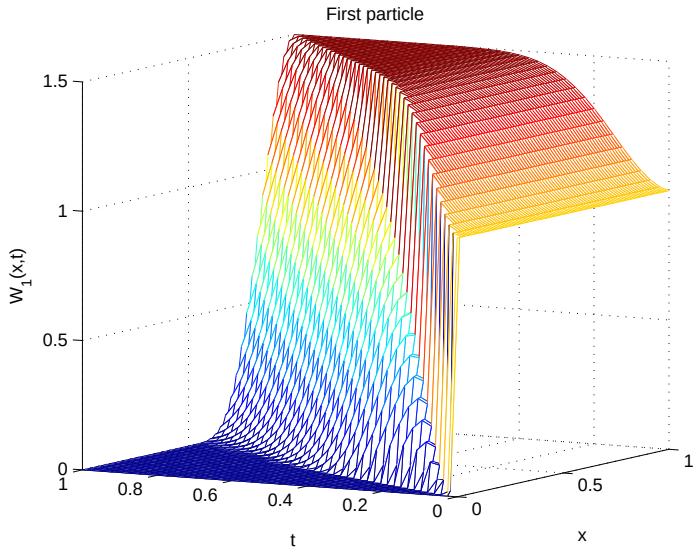

Figure 10: Result for the first particle.

```
A =
     1     2     3
     4     5     6
     7     8     9
```

The following examples show how to access the elements of matrices:

```
>> A(2,3)
ans =
     6
>> A(:,2)
ans =
     2
     5
     8
>> A(3,:)
ans =
     7     8     9
```

To transpose a matrix or a vector use ′:

```
>> A'
ans =
     1     4     7
```

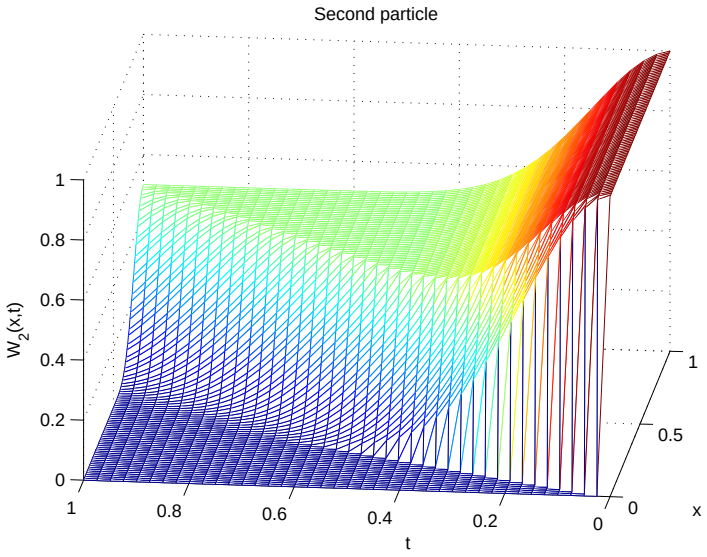

Figure 11: Result for the second particle.

```
2     5     8
3     6     9
```

It is often useful, when entering a matrix, to suppress the display; this is done by ending the line with a semicolon.

Graphs and Plots

The simplest graphs to create are plots of points in the cartesian plane. For example:

```
>> x = [1;2;3;4;5];
>> y = [0;.25;3;1.5;2];
>> plot(x,y)
```

In order to create a graph of a surface in 3-space, it is necessary to evaluate the function on a regular rectangular grid. This can be done using the *meshgrid* command. First, create $1 - D$ vectors describing the grids in the x- and y-directions:

```
>> x = (0 : 2*pi/60 : 2*pi)';
>> y = (0 : 4*pi/120 : 4*pi)';
```

Next, "spread" these grids into two dimensions using *meshgrid*:

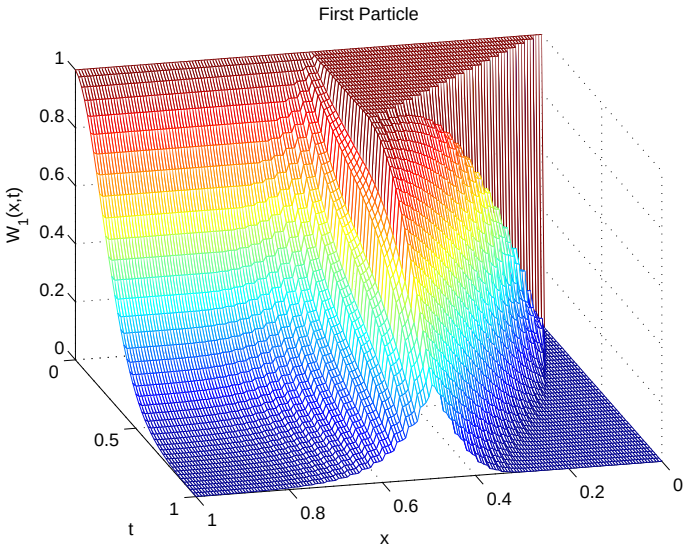

Figure 12: Result for the first particle.

```
>> [X,Y] = meshgrid(x,y);
```

The effect of *meshgrid* is to create a vector X with the x-grid along each row, and a vector Y with the y-grid along each column. Then, using vectorized functions and operators, it is easy to evaluate a function $z = f(x, y)$ of two variables on the rectangular grid:

```
>> z = cos(X).*cos(2*Y);
```

Loops

Matlab provides two types of loops, a for-loop and a while-loop. A for-loop repeats the statements in the loop as the loop index takes on the values in a given row vector:

```
>> for i = [1,2,3,4]
i^2
end
```

The loop must be terminated by end. The while-loop repeats as long as the given expression is true (nonzero):

```
>> x = 1;
>> while (1+x) > 1
x = x/2;
```

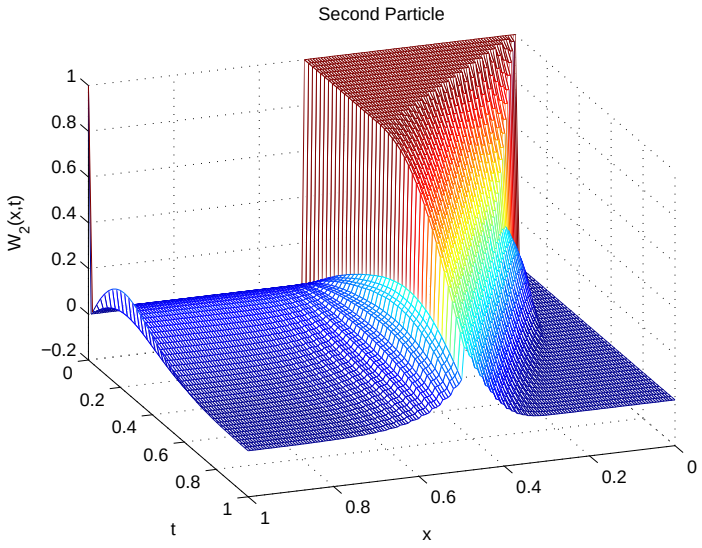

Figure 13: Result for the second particle.

```
end
>> x
x =
  1.1102e-016
```

Functions in Matlab

In Matlab the common scientific functions, such as square root, sine, cosine, tangent, exponential, and logarithm are predefined. Moreover, you can define a function in an m-file that begins with a line of the following form:

```
function [output1,output2,...] = cmd_name(input1,input2,...)
```

Here is a simple example of a function; it computes the function $y = sin(x^2)$. The following commands should be stored in the file myFunction.m (the name of the function within Matlab is the name of the m-file, without the extension):

```
function y = myFunction(x)
y = sin(x.^2);
```

With this function defined, we can now use myFunction just as the built-in function sin:

```
>> x = (-pi : 2*pi/100 : pi)';
>> y = sin(x);
```

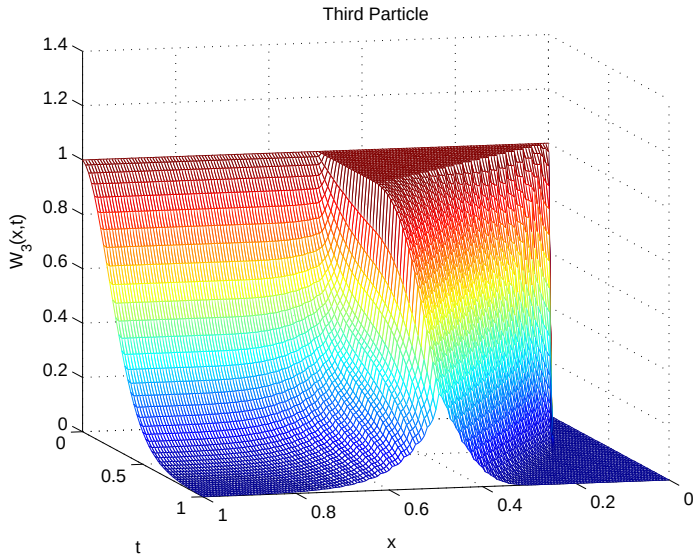

Figure 14: Result for the third particle.

```
>> z = myFunction(x);
>> plot(x,y,'r',x,z,'b')
```

7.2 Matlab Codes

Program-Code for the ODE system

The analytical solver method for the homogeneous ODE-system (5 particles):

```
k12=0.2;
k13=0.3;
k14=0.2;
k15=0.7;
k21=0.1;
k23=0.8;
k24=0.9;
k25=0.3;
k31=0.1;
k32=0.7;
k34=0.9;
k35=0.9;
k41=0.8;
k42=0.9;
k43=0.1;
```

```
k45=0.2;
k51=0.1;
k52=0.2;
k53=0.3;
k54=0.4;
A=[(-k12-k13-k14-k15), k21, k31, k41, k51;
   k12, (-k21-k23-k24-k25), k32, k42, k52;
   k13, k23, (-k31-k32-k34-k35), k43, k53;
   k14, k24, k34, (-k41-k42-k43-k45), k54;
   k15, k25, k35, k45, (-k51-k52-k53-k54)];
yo=[1.0/5.0; 1.0/5.0; 1.0/5.0; 1.0/5.0; 1.0/5.0;];
to=0.0;
tf=10.0;
n=max(size(A));
[wcol, lambdac]=eig(A);
wcolinv = inv(wcol);
npoints = 1000;
dt = (tf-to)/npoints;
for i = 1:1:npoints+1
    tp(i) = (i-1)*dt +to;
end
explambda = zeros(n,n);yp = zeros(n,npoints);
for i = 1:1:npoints+1
        for j = 1:n
             explambda(j,j) = exp(lambdac(j,j)*(tp(i)-to));
        end
yp(:,i) = wcol*explambda*wcolinv*yo;
end
         plot(tp, yp(1,:), 'g-'), xlabel( 't' ), ylabel ( 'y' )
         hold on
         plot(tp, yp(2,:), 'r-'), xlabel( 't' ), ylabel ( 'y' )
         hold on
         plot(tp, yp(3,:), 'b-'), xlabel( 't' ), ylabel ( 'y' )
         hold on
         plot(tp, yp(4,:), 'k-'), xlabel( 't' ), ylabel ( 'y' )
         hold on
         plot(tp, yp(5,:), 'm-'), xlabel( 't' ), ylabel ( 'y' )
hold off
```

Program-Code for the PDE system

Finite difference method for the PDE-system (5 particles):

```
function transport_characteristic

 for k = 1 : 5
```

```
    l = 1/(2^(k-1));  %faktor

    T = 1;
    M = 80;
    dx = 1/M;
    dt = 1/(l*80);
    N = floor(T/dt);
    mu = l*dt/dx;
    U = zeros(N+1,M+1);


    for j = 1 : M+1
        U(1,j) = phi((j-1)*dx);
    end
    plot([0:dx:1],U(1,:)); axis([0 1 0 1]); pause

    for n = 1 : N
        U(n+1,1) = 0;
        for j = 2 : M + 1
            U(n+1,j) = U(n,j) - mu * (U(n,j) - U(n,j-1));
        end
        plot([0:dx:1],U(n+1,:)); axis([0 1 0 1]); pause(.1)
    end

    mesh(0:dx:1,0:dt:T,U); pause
    title('Approximation of u_t + u_x = 0 with finite differences');
    xlabel('x'); ylabel('t'); zlabel('U(x,t)');
end


function y = phi(x)
 if x < .5
     y = .5 * (1 + sin(4*pi*x-pi/2));
 else
     y = 0;
 end
```

Program-Code for the coupled version

PDE and ODE Solver (2 particles):

Program-Code for the coupled version

PDE and ODE Solver (3 particles):

```
function Coupled_3_Particles


%parameters

T = 1;
N = 40;
M = 40;
dt = T/N;
dx_1 = 1/M;
dx_2 = 1/(2*M);
dx_3 = 1/(4*M);function Coupled_2_Particles


%parameters

T = 1;
N = 40;
M = 40;
dt = T/N;
dx_1 = 1/M;
dx_2 = 1/(2*M);
U = zeros(N+1,M+1);
Z = zeros(M+1,(2*M)+1);
V_1 = zeros(N+1,(2*M)+1);
V_2 = zeros(N+1,(2*M)+1);
W_1 = zeros(N+1,(2*M)+1);
W_2 = zeros(N+1,(2*M)+1);

k11=0.25;
k12=0.75;
k21=0.25;
k22=0.75;
A=[-k11, k12; k21, -k22];
mx=max(size(A));
[wcol, lambdac] = eig(A);
wcolinv = inv(wcol);
explambda = zeros(mx,mx);
yp = zeros(mx,mx,N+1);


%reactionmatrix

for i = 1 : N+1
        for j = 1 : mx
             explambda(j,j) = exp(lambdac(j,j)*((i-1)*dt));
```

```
        end
yp(:,:,i) = wcol*explambda*wcolinv;

end


%prolongation

for i = 1 : M
    Z(i,2*i) = 1;
end

for i = 1 : M+1
    Z(i,(2*i)-1) = 1;
end
Z(M+1, (2*M)+1) = 1;


%coupling

     for j = 1 : M+1
         U(1,j) = phi((j-1)*dx_1);
     end
     V_1 = U*Z;

     for j = 1 : (2*M)+1
         V_2(1,j) = phi((j-1)*dx_2);
     end

     for i = 1:2*M+1
        W_1(1,i) = V_1(1,i)*yp(1,1,1) + V_2(1,i)*yp(1,2,1);
     end
     plot([0:dx_2:1],W_1(1,:)); axis([0 1 0 1]); pause(.05)

     for i = 1:2*M+1
        W_2(1,i) = V_1(1,i)*yp(2,1,1) + V_2(1,i)*yp(2,2,1);
     end
     plot([0:dx_2:1],W_2(1,:)); axis([0 1 0 1]); pause(.05)


for n = 1:N

     U(n+1,1) = 0;
     for j = 2:M+1
         U(n+1,j) = W_1(n,2*(j-1));
     end
```

```
     V_1 = U*Z;

     V_2(n+1,1) = 0;
     for j = 2 : 2*M+1
         V_2(n+1,j) = W_2(n,j-1);
     end


     for i = 1:2*M+1
        W_1(n+1,i) = V_1(n+1,i)*yp(1,1,n+1) + V_2(n+1,i)*yp(1,2,n+1);
     end
     plot([0:dx_2:1],W_1(n+1,:)); axis([0 1 0 1]); pause(.05)

     for i = 1:2*M+1
        W_2(n+1,i) = V_1(n+1,i)*yp(2,1,n+1) + V_2(n+1,i)*yp(2,2,n+1);
     end
     plot([0:dx_2:1],W_2(n+1,:)); axis([0 1 0 1]); pause(.05)

 end


%graphics

mesh(0:dx_2:1,0:dt:T,W_1);
title('First particle');
xlabel('x'); ylabel('t'); zlabel('W_1(x,t)'); pause


mesh(0:dx_2:1,0:dt:T,W_2);
title('Second particle');
xlabel('x'); ylabel('t'); zlabel('W_2(x,t)');



function y = phi(x)
 if 0 <= x <= 1
     y = 1;
 else
     y = 0;
 end
U_1 = zeros(N+1,M+1);
U_2 = zeros(N+1,(2*M)+1);
Z_1 = zeros(M+1,(4*M)+1);
Z_2 = zeros((2*M)+1,(4*M)+1);
V_1 = zeros(N+1,(4*M)+1);
V_2 = zeros(N+1,(4*M)+1);
```

```
V_3 = zeros(N+1,(4*M)+1);
W_1 = zeros(N+1,(4*M)+1);
W_2 = zeros(N+1,(4*M)+1);
W_3 = zeros(N+1,(4*M)+1);

k12=0.5;
k21=0.5;
k13=0.5;
k31=0.5;
k23=0.5;
k32=0.5;
A=[(-k13-k12), k21, k31; k12, (-k21-k23), k32; k13, k23, (-k31-k32)];
mx=max(size(A));
[wcol, lambdac] = eig(A);
wcolinv = inv(wcol);
explambda = zeros(mx,mx);
yp = zeros(mx,mx,N+1);

Y_1 = zeros(1,N+1);
Y_2 = zeros(1,N+1);


%reactionmatrix

for i = 1 : N+1
        for j = 1 : mx
             explambda(j,j) = exp(lambdac(j,j)*((i-1)*dt));
        end
yp(:,:,i) = wcol*explambda*wcolinv;
end


%prolongations

for k = 1:4
    for i = 1 : M
        Z_1(i,(4*i)-(k-1)) = 1;
    end
end
Z_1(M+1, (4*M)+1) = 1;

for k = 1:2
    for i = 1 : M
        Z_2(i,2*i-(k-1)) = 1;
    end
end
```

```
Z_2((2*M)+1, (4*M)+1) = 1;


%coupling

for j = 1 : M+1
    U_1(1,j) = phi((j-1)*dx_1);
end
V_1 = U_1*Z_1;

for j = 1 : (2*M)+1
    U_2(1,j) = phi((j-1)*dx_2);
end
V_2 = U_2*Z_2;

for j = 1 : (4*M)+1
    V_3(1,j) = phi((j-1)*dx_3);
end

for i = 1:(4*M)+1
    W_1(1,i) = V_1(1,i)*yp(1,1,1) + V_2(1,i)*yp(1,2,1) + V_3(1,i)*yp(1,3,1);
end
plot((0:dx_3:1),W_1(1,:)); axis([0 1 0 1]); pause(.05)

for i = 1:(4*M)+1
    W_2(1,i) = V_1(1,i)*yp(2,1,1) + V_2(1,i)*yp(2,2,1) + V_3(1,i)*yp(2,3,1);
end
plot((0:dx_3:1),W_2(1,:)); axis([0 1 0 1]); pause(.05)

for i = 1:(4*M)+1
    W_3(1,i) = V_1(1,i)*yp(3,1,1) + V_2(1,i)*yp(3,2,1) + V_3(1,i)*yp(3,3,1);
end
plot((0:dx_3:1),W_3(1,:)); axis([0 1 0 1]); pause(.05)


for n = 1:N

    U_1(n+1,1) = 0;
    for j = 2 : M+1
        U_1(n+1,j) = W_1(n,4*(j-1));
    end
    V_1 = U_1*Z_1;

    U_2(n+1,1) = 0;
    for j = 2 : 2*M+1
        U_2(n+1,j) = W_2(n,2*(j-1));
```

```
    end
    V_2 = U_2*Z_2;

    V_3(n+1,1) = 0;
    for j = 2 : (4*M)+1
        V_3(n+1,j) = W_3(n,j-1);
    end


    for i = 1:(4*M)+1
    W_1(n+1,i) = V_1(n+1,i)*yp(1,1,n+1) + V_2(n+1,i)*yp(1,2,n+1) + V_3(n+1,i)*yp(1,3,n+1
    end
    plot((0:dx_3:1),W_1(n+1,:)); axis([0 1 0 1]); pause(.05)

    for i = 1:(4*M)+1
    W_2(n+1,i) = V_1(n+1,i)*yp(2,1,n+1) + V_2(n+1,i)*yp(2,2,n+1) + V_3(n+1,i)*yp(2,3,n+1
    end
    plot((0:dx_3:1),W_2(n+1,:)); axis([0 1 0 1]); pause(.05)

    for i = 1:(4*M)+1
    W_3(n+1,i) = V_1(n+1,i)*yp(3,1,n+1) + V_2(n+1,i)*yp(3,2,n+1) + V_3(n+1,i)*yp(3,3,n+1
    end
    plot((0:dx_3:1),W_3(n+1,:)); axis([0 1 0 1]); pause(.05)

end


%graphics

mesh(0:dx_3:1,0:dt:T,W_1(:,:));
title('First Particle');
xlabel('x'); ylabel('t'); zlabel('W_1(x,t)'); pause

mesh(0:dx_3:1,0:dt:T,W_2(:,:));
title('Second Particle');
xlabel('x'); ylabel('t'); zlabel('W_2(x,t)'); pause

mesh(0:dx_3:1,0:dt:T,W_3(:,:));
title('Third Particle');
xlabel('x'); ylabel('t'); zlabel('W_3(x,t)');


%mass-conservation

for j = 1 : N+1
    Y_1(1,j) = dx_1*1/2*(sum(V_1(j,1:4*M)) + sum(V_1(j,2:4*M+1))) + dx_2*1/2*(sum(V_2(j,
```

```
end

Y_1

sum(Y_1(1,1:N+1))

for j = 1 : N+1
    Y_2(1,j) = dx_1*1/2*(sum(W_1(j,1:4*M)) + sum(W_1(j,2:4*M+1))) + dx_2*1/2*(sum(W_2(j,
end

Y_2

sum(Y_2(1,1:N+1))



function y = phi(x)
 if 0 <= x <= 1
     y = 1;
 else
     y = 0;
 end
```